第42卷 第5期
2022年10月

桂林电子科技大学学报
Journal of Guilin University of Electronic Technology

Vol. 42, No. 5
Oct. 2022

面向检索服务的词干提取与相关排序优化研究

朱 艳[1]，张敬伟[1]，杨 青[2]，胡晓丽[1]，单美静[3]

（1.桂林电子科技大学 广西可信软件重点实验室，广西 桂林 541004；
2.桂林电子科技大学 广西自动检测技术与仪器重点实验室，广西 桂林 541004；
3.华东政法大学 刑事法学院，上海 201620）

摘 要：新一代信息技术的兴起以及互联网产业的飞速发展使得数据量呈爆炸式增长。为满足数十亿用户从海量数据中快速获取有效信息的需求，提升搜索引擎的检索质量以及查询效率具有重要意义，同时也面临挑战。一方面，用户的查询词日益复杂，语言词汇形态变异的特点导致检索词变得多样化，而现有词干提取算法普遍存在词干提取不足、词干提取准确率不高等问题；另一方面，在海量数据中检索到满足用户查询要求的文档结果是一项非常耗时的任务，而现有将文档划分到多个服务器处理查询延迟的方法常常会出现尾延迟问题。针对以上问题，在文本预处理阶段，设计了词形规范化算法APS，对规则函数进行重编码，优化了特征词提取；在相关排序阶段，设计了基于一次一得分查询处理策略的随时排序算法SAR，在给定时间预算处理完指定数量倒排段后能够提前终止查询过程，大大减少了查询评估时间。在多个真实数据集上进行了实验，验证了APS算法对于提高词干提取准确率的有效性以及SAR算法对于控制查询延迟的真实性。

关键词：词干提取算法；随时排序算法；文本预处理；SAAT；相关排序

中图分类号：TP391 **文献标志码**：A **文章编号**：1673-808X(2022)05-0354-12

Research on stemming and related ranking optimization for retrieval service

ZHU Yan[1], ZHANG Jingwei[1], YANG Qing[2], HU Xiaoli[1], SHAN Meijing[3]

(1. Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China;
2. Guangxi Key Laboratory of Automatic Detection Technology and Instrument, Guilin University of Electronic Technology, Guilin 541004, China;
3. School of Crimincd Law, East China University of Political Science and Law, Shanghai 201620, China)

Abstract: The rise of a new generation of information technology and the rapid development of the internet industry have led to an explosive growth in the amount of data. In order to meet the needs of billions of users to obtain effective information from massive data quickly, it is of great significance to improve the retrieval quality and query efficiency of search engines, but it also faces challenges. On the one hand, the query words of users are becoming more and more complex, and the characteristics of the morphological variation of language vocabulary lead to the diversification of search words, while existing stemming algorithms generally suffer from under stemming and unsatisfactory stemming accuracy; On the other hand, it is a very time-consuming task to retrieve document results that meet user query requirements from massive data, and existing methods of dividing documents into multiple servers to handle query latency often suffer from tail latency problems. In view of the above problems, in the text preprocessing stage, the word form normalization algorithm APS (advanced porter stemmer) is designed, the rule function is recoded, and the feature word extraction is optimized; In the related ranking stage, the anytime ranking algorithm SAR (SAAT anytime ranking) is designed based on the score-at-a-Time query processing strategy, which can terminate the query process in advance after a given time budget or processing a specified number of inverted segments and control the query delay effectively. Experiments are carried out on multiple real datasets to verify the effective-

收稿日期：2022-04-19

基金项目：国家自然科学基金(61862013，U1811264，U1711263)；广西自然科学基金(2020GXNSFAA159117，2018GXNS FAA281199)；广西可信软件重点实验室重点基金(KX202052)；广西自动检测技术与仪器重点实验室主任基金(YQ21102)

通信作者：单美静(1979－)，女，教授，博士，研究方向为大数据分析、信息安全等。Email：shanmeijing@ecupl.edu.cn

引文格式：朱艳，张敬伟，杨青，等.面向检索服务的词干提取与相关排序优化研究[J].桂林电子科技大学学报，2022，42(5)：354-365.

ness of the APS algorithm in improving the accuracy of stemming and the authenticity of the SAR algorithm in controlling query latency.

Key words: stemming algorithms; anytime ranking; text preprocessing; SAAT; related ranking

云计算、大数据等新技术的兴起，以及电子商务、网络自媒体、娱乐通讯等互联网产业的蓬勃发展使得信息量呈现指数级增长。据统计[1]，全球每年产生的数据量高达 1～2 EB，其中非纸质信息就占了 99.7%。尽管大数据技术、深度学习以及神经网络计算能力的进步加速了信息处理能力的提升，但对信息过载问题的缓解仍旧微乎其微。在关注度有限的情况下，如何短时间内从指数级增长的数据中获取有效信息成为了亟待解决的问题，而搜索引擎则是人们提取信息的有效方式之一。

随着互联网行业的快速发展，搜索用户的信息需求日益复杂，同时检索词也逐渐变得多样化，一个词常有多种不同形态，这些都对语料库学习的准确度产生一定影响。研究表明[2]，若检索词未进行词形规范化，可能会造成重要的检索结果缺失或存在过多无关的文档出现在检索结果列表的情况，而若检索词为主题词表中的词语，则能有效提高检索结果的准确率与查全率。因此，在信息检索与文本挖掘研究中，需要对单词进行归一化处理，以提高文本处理的效率，其中词干提取是词形归一化的核心技术之一。然而现有的词干提取算法普遍存在词干提取不足、词干提取准确率不高等问题，无法有效改善庞大的文本词汇量与关键词特征缺失的矛盾问题，导致搜索引擎的时空复杂度偏高而查询效率偏低。为解决文本查询处理面临的“高维-稀疏”问题，通过优化词干分析算法对文本向量空间进行降维处理，以减少词项的数量，从而提高文本处理效率。

此外，为了减少系统在相关排序过程中的时间及硬件资源消耗，查询优化技术逐渐受到学术界及工业界的重视。其中，top-k 查询排序是信息检索领域广泛应用的查询处理优化技术之一。相关文档 top-k 排序基于查询-文档的相似度得分，以及具体的得分聚合函数从海量文本数据中返回 k 个最大的得分排名结果。现有的 top-k 排序研究大多是确定了整体的 top-k 结果后，才会停止排序过程。尽管这种方式通过详尽遍历所有文档和词项能够保证检索质量，但同时对海量文档的处理也产生了不可忽视的查询延迟。研究表明[3-4]，响应时间过长直接影响用户体验，造成潜在利益的巨大损失。目前对于查询延迟的处理，大多通过将文档集合划分到若干服务器来管理，但这种方式仍存在尾延迟[5-11]的问题。对于大规模分布式系统来说，尾延迟现象更加普遍，甚至会严重影响服务的整体性能。而随时排序算法能够在给定时间预算内或给定倒排段处理数量下，随时停止检索过程，从而控制查询延迟。因此，当存在一定查询负载时，利用随时排序算法能够大大降低整个系统的资源损耗及维护成本，解决普遍存在的高百分比尾延迟问题[12]，以适应服务水平协议对响应时间的要求。

基于对上述问题的思考，在文本预处理与相关排序 2 个方面进行了深入研究：

首先，在文本预处理阶段，设计了词形规范化算法（advanced porter stemmer，简称 APS），解决了现有算法存在的词干提取不足、词干提取准确率高等问题。该算法基于屈折派生形态学调整了规则函数的定义，优化了特征词提取，并且补充了不规则动词以及若干后缀的处理，同时添加了对停用词过滤的支持。针对 APS 算法的评价，在 3 个真实的数据集上开展实验，验证了 APS 优化算法对于解决词干不足问题的有效性以及提高词干提取准确率的真实性。

其次，在相关排序阶段，设计了基于一次一得分（score-at-a-time，简称 SAAT）查询处理策略的随时排序算法（SAAT-anytime ranking，简称 SAR）。该算法能够在处理完指定数量的倒排段后或给定时间预算内提前终止查询过程，大大减少了查询评估延迟时间，在牺牲可接受范围内检索质量的情况下，能够返回较为准确的检索结果，解决了现有方法普遍存在的尾延迟问题。在 2 个真实的大型 TREC 标准数据集 ClueWeb09b 和 ClueWeb12-B13 上进行了实验，通过检索质量评价指标 nDCG@10 对 SAR 算法进行了评估，并记录了在给定时间预算下的查询延迟、减少的倒排段处理数量，验证了 SAR 算法对于控制尾部延迟时间的有效性。

1　相关工作

近年来，搜索引擎的优化问题已被广泛研究。在互联网信息量以指数级增长，信息过载问题愈发严峻的时代背景下，如何尽快找到满足用户需求的文档内容，提高信息检索的效率日益成为研究者关注的焦点问题，这也为科学研究提供了动力。本节将主要围绕词干提取与相关查询 2 个方面对以往工作进行总结概括。

1.1 词干提取

根据词干提取方法的实现原理，可以将其归为 4 类：基于规则的词缀删除方法[13-17]、基于词典查找的方法[18]、基于单词分布规律的统计方法[19-21]以及混合方法[22-24]。基于词典查找的方法在权威词典的支持下，结果更加准确，能够处理部分不规则变换词，但遍历词典查找费时且对词典具有依赖性。基于统计的方法主要是针对词典中未收录的词以及不规则变化词，通过统计单词规律对单词进行规范化，因此不受语种限制，但识别出的词干误差较大，且准确率不稳定。二者更适用于对小语种单词的词干提取。而混合型方法虽然融合了多种方法的优势，词干提取的准确率更高，但算法流程复杂，需要考虑的因素过多，且需要多种背景知识的支持，因此限制较大，效率较低。而基于规则的词缀删除方法能够快速处理常规词的变换，适用范围更广。因此，主要针对基于规则的词缀删除方法进行改进优化。

基于规则的词缀删除方法利用单词屈折派生形态中具备的内在规律，对单词中的词缀进行处理。1968 年，Lovins[13]提出了有效的同名词干提取 Lovins 算法，该算法基于最长匹配原则对照词缀列表去除单词后缀后，匹配规则列表中的转换规则，重新对单词进行编码，将词干转换为有效单词，最终提取出词干。其优点是规则简单，且能够处理某些叠词结尾的单词以及不规则单词复数；但缺点是非常耗时，且词干提取的准确率不高。针对 Lovins 算法的规则和匹配方法存在的不足，Dawson[14]提出了同名方法 Dawson 算法。该算法基于部分匹配的思想，在限制条件下匹配相同词干，扩展了 Lovins 算法，并解决了拼写异常问题。Dawson 算法是单程非迭代算法，因此执行速度快，但该算法的缺点是复杂，且缺乏标准的可重用实现。Lancaster(Paice/Husk)[15]算法是一种迭代算法，通过判断是否需要再次提取词干循环执行匹配流程。该算法通过将单词最后一个字符作为索引寻找适用规则，每条规则决定是否对后缀进行删除或替换，若规则不匹配或满足词干提取结束条件，则终止流程，输出词干。Lancaster 算法的优点是，每次迭代都会应用规则进行删除和替换，降低了词干提取不足的概率；但缺点是算法繁杂，可能会出现词干过度提取的情况。Porter Stemmer(波特词干)[16-17]算法自提出以来便广受欢迎，现已广泛应用于信息检索领域以及多种检索系统中，如 Lucene、Solr 等。波特词干算法对许多基本算法进行了改进和优化，主要用于对英文单词中通用形态以及屈折词缀进行剔除。尽管该算法在多种算法基础上做出了改进，但缺乏对不规则动词、不规则名词复数以及多种词缀的考虑，因此仍存在词干提取不足以及词干提取准确率不高等问题，需进一步优化。

1.2 相关排序

将文档数据与查询信息进行预处理后，需要对文档和查询的相关度进行计算，进而根据得分高低对相关文档进行排序，最后返回给用户得分 top-k 的文档结果，这个排序的过程称为相关排序。目前搜索引擎的排序策略往往建立在所有文档的相关度得分上，然而穷尽处理所有候选结果所花费的时间和资源开销过大。在当下互联网的数据规模以指数级增长的背景下，为了提升查询性能，相关优化技术不断推陈出新。目前主流的查询效率优化技术包括剪枝算法、选择搜索以及随时排序算法等。

动态剪枝算法以处理尽可能少的相关文档为目标，采用跳跃式访问倒排列表的方式来减少对无关或相关度较低的文档的处理，避免对所有文档的遍历和访问，从而提高查询效率。动态剪枝算法能够保证 top-k 个文档列表的计算是安全的，也就是说使用动态剪枝算法与穷尽查询方法得到的查询结果相同。常用的动态剪枝算法有 MaxScore[25]、WAND[26]、BMW[27-28]以及 VBMW[29]等。但有研究表明[30]，剪枝算法执行尾部查询所花费的时间比平均查询延迟时间要多若干数量级。

选择搜索在搜索构建时，将文档集合按照主题划分，理想情况下每个分片都包含一组主题相关的文档[31-32]。传入的每个用户查询都由代理流程预测被划分的集合分片，然后由划分的分片处理查询，最后将分片结果汇总。每个分片的处理过程都能应用动态剪枝算法。该方法的优点是，能够有效减少工作负载，查询效率高；但缺点是，由于只有部分分片对查询进行处理，算法得到的结果可能会与穷尽查询算法得到的结果有所偏差。

随时排序算法实现基于影响力排序的索引(impact-ordered index)。相对于一次一文档(term-at-a-time)查询处理策略，SAAT 查询策略能够根据影响力得分来处理文档的优先级[32-33]，可在避免遍历所有文档的情况下，输出较为准确的排序结果，更有利于提前终止文档相关度计算流程，这与随时排序的目标相同，因此随时排序算法大都基于 SAAT 策略。当响应时间预先由服务水平协议确定时，查询处理过程必须支持可中断，随时排序算法针对此类情况给出了解决方案。随时排序算法在给定时间预算内返回尽

可能准确的结果,且检索结果质量随着预算时间的延长而成正比提升[34-35]。基于以上理论,在相关排序阶段通过设计基于 SAAT 策略的随时排序算法来控制查询延迟时间。

2 基于改进 Porter Stemmer 的 APS 算法

针对 Porter Stemmer 存在的词干提取不足以及词干提取准确率不高等问题,对波特词干算法进行改进,设计了 APS 算法。该算法重新编码了规则函数,优化了特征词提取,并补充了不规则动词以及若干后缀的处理,同时添加了对停用词过滤的支持。

为使算法描述更清晰,首先对以下定义进行说明:

定义 1 元音(Vowel)。a,e,i,o,u 五个字母。

定义 2 辅音(Consonant)。除元音外的其他字母。

定义 3 给定单词 T,以词缀 S_1 结尾,若词干满足指定条件 condition,则由新词缀 S_2 代替 S_1,即:

$$(\text{condition})S_1 \rightarrow S_2。\tag{1}$$

定义 4 屈折形态(Inflexion)。单词或词根受语法影响,加上屈折词缀后的形态,包括单词复数形式如“apples”等、不同时态形式如“looked”等、以及分词形式如“walking”等。

定义 5 派生形态(Morphological Derivation)。单词或词根在句法范畴基础上,添加实质性的词缀后所派生的形态,如 illegal,irregular 等。

定义 6 叠字(Double)。由单个字母重叠而成的词缀,如 tt、mm、nn 等。

定义 7 复合词缀(Double Suffix)。由多个词缀整合而成的形态,如由 general 附加 ize 后缀和 ation 后缀整合得到 generalization,其中 generalization 的词缀为复合词缀。

APS 算法基于英文单词形态特征及屈折派生形态学,针对波特词干算法存在的不足,做以下优化:

1)对不规则动词变位与复数的特例进行补充。波特词干算法忽略了 2 种不规则词形式的处理:①不符合任何特征规则的动词,例如单词“buy”及其过去式“bought”。对于此类情况,通过枚举不规则动词形式进行改善;②符合一般规则特征的单词,例如以-foot 结尾的单词复数形式以-feet 结尾。对于此类情况,通过添加对规则的补充可以得到改善。表 1 为波特词干算法与 APS 算法处理前后的对照示例 1。

表 1 波特词干算法与 APS 算法处理对照示例 1

原词	波特词干	APS	规则
child/children	child/children	child/child	children→-childr
knife/knives	knif/kniv	knif/knif	-knives→knife
aviatrix/aviatrices	aviatrix/aviatric	aviatrix/aviatrix	-trices→trix

2)对以-s 结尾的动词及其分词形式的处理进行优化。波特词干算法对于以-s 结尾的动词分词形式的处理方式是直接去除末尾的-ed 或-ing,保留末尾的-s。在该规则下,对于“focus”与其复数“focuses”,存在将“focuses”转化为词干“focus”,而将“focus”转化为“focu”的错例。针对此类情况,通过优化规则可以改善:若以-s 结尾,但不以 ss 结尾的单词,一律转化为 s。表 2 为波特词干算法与 APS 算法处理前后的对照示例 2。

表 2 波特词干算法与 APS 算法处理对照示例 2

原词	波特词干	APS	规则
chorus/chorused/chorusing	choru/chorus/chorus/chorus	choru/choru/choru/choru	-sed && ! (-ssed)→s -sing && ! (-ssing)→s
focus/focuses/focused/focusing	focu/focus/focus/focus	focu/focu/focu/focu	

3)对以-y 结尾单词的词干合并方式进行优化。波特词干算法对于以-y 结尾的单词的处理方式是:若包含元音,则将-y 转变为-i;另外,针对以-ies 结尾的单词处理方式是:将 ies 转变为 i。这种规则能正确处理包含元音的单词,例如 carry→carries,marry→marries 等。但对于不包含元音的词干则不适用,例如 cry-cries-cried,则会被转化为 cry-cri-cri-cry。同理,以-ye 结尾的单词也不适用,因为末尾的 e 最终会去除。针对此类情况,通过优化规则:首先将分词后缀-es/-ed/-ing 去除,然后删除规则“若包含元音,则将末尾的 y 转变为 i”,即保持末尾的-y 不变。表 3 为波特词干算法与 APS 算法处理前后的对照示例 3。

表 3 波特词干算法与 APS 算法处理对照示例 3

原词	波特词干	APS
try/tries/tried/trying	try/tri/tri/try	cry/cry/cry/cry
dye/dyes/dyed/dying	dye/dyes/dyed/dying	dy/dy/dy/dy

4)对以双辅音结尾的单词及其衍生词的处理进行优化。波特词干算法对于以非‘l’、‘s’或‘z’双辅音结尾单词的分词形式处理方式是:去除一个辅音,保留一个辅音。在这种规则下,会出现错将单词“ebbed”转换为“eb”,而“ebb”转换为“eb”的错误案例。另外,若存在以-z 结尾的单词,但其分词加了叠

词词缀即-zz，例如单词“whiz”的过去分词“whizz”，“whiz”本身会转化为“whiz”，而过去分词“whizz”则转化为“whiz”，误判情况出现。针对以上情况，可优化规则：删除所有以除-l 双辅音结尾单词的辅音字母，对于以双辅音-ll 结尾的单词，若 $m>1$，则删除一个辅音。表 4 为波特词干算法与 APS 算法处理前后的对照示例 4。

表 4 波特词干算法与 APS 算法处理对照示例 4

原词	波特词干	APS
add/added/adding	add/ad/ad	ad/ad/ad
staff/staffed/staffing	staff/staf/staf	staf/staf/staf
whiz/whizzes/whizzed	whiz/whizz/whizz	whiz/whiz/whiz

5）对部分现在分词以及过去分词衍生词的处理进行优化；波特词干算法忽略了对现在分词、过去分词衍生词的处理，例如“study”转化为 “studi”，而“studiedly”却转化为“studiedli”。对于该类情况的处理，APS 补充了对该类词的转化规则。表 5 为波特词干算法与 APS 算法处理对照示例 5。

表 5 波特词干算法与 APS 算法处理对照示例 5

原词	波特词干	APS	规则
amaze/amazed/amazedly/amazedness	amaz/amaz/amazedli/amazed	amaz/amaz/amaz/amaz	-ly→-ed -ss→ed
study/studied/studiedness/studiedly	studi/studi/studied/studiedli	study/study/study/study	-ly→ied -ss→-ied

6）补充了若干后缀的处理。针对波特词干算法忽略-tor、-sory、-ship 等若干词缀，APS 算法进行了补充。另外对于单词的复合后缀的漏判问题，通过由后缀枚举所有可能的复合后缀进行优化。例如，由词缀-ate 衍生出的-ative、-atic 等词缀都将被对应到词缀-ate。表 6 为部分词缀转换示例。

表 6 APS 算法词缀转换示例

词缀	转换规则
-atization	-atization→ate
-atist	-atist→-ate
-atism	-atism→-ate

APS 算法进行词干提取的整体流程如图 1 所示。由图 1 可知，APS 算法对词干的提取主要包括 5 个步骤：第一步，处理单词的屈折形态，包括单词的复数、现在分词、过去分词等，例如将“apples”转换为“apple”，将“looked”转换为“look”；第二步，根据前文描述的优化工作对 y→i 的规则进行重编码，例如将“try”转换为“tri”；第三步，对整合多个词缀的复合词缀进行处理，将这类词缀转化为非复合后缀，例如将“generalization”转换为“generalize”。本算法对复合词缀到非复合后缀的映射规则进行了重编码；第四步，删除简单的非复合后缀，通过定义的编码规则对现存词干进行归一化，例如将上一步得到的“generalize”转换为“general”。这两步主要对单词的派生形态进行处理。第五步，处理不满足以上编码规则的不规则词，通过与补充的规则转化表单词进行遍历匹配来完成对不规则词的词干提取；最后，在处理完不规则词的基础上，根据重编码后的新规则去除单词末尾的-e 或-l，最终得到词干。

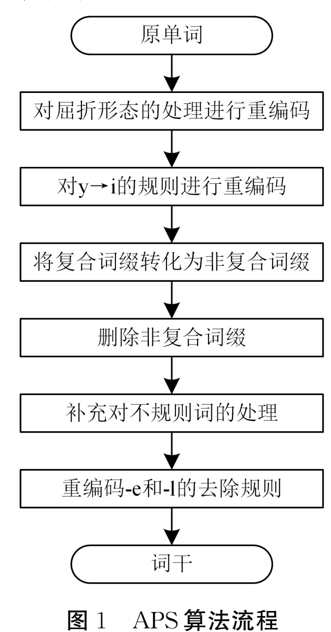


图 1 APS 算法流程

3 基于 SAAT 策略的随时排序算法 SAR

搜索引擎在海量数据中检索到满足用户查询要求的文档是一项非常耗时的任务。研究表明[36]，在谷歌搜索中人为对查询时间延长 100～400 ms，用户每天的搜索次数减少 0.2%～0.6%。现有的处理查询延迟的方法往往是将文档划分到多个服务器，每个服务器分担部分时间延迟，但查询的延迟时间仍不可忽视。基于对提升用户体验的考虑，分析发现，通过牺牲可接受范围的搜索质量能够在任意给定时间限制的情况下，向用户查询返回较为准确的文档排名，并且随着计算时间的延长，结果质量成正比增长。在此基础上，基于 SAAT 查询处理策略设计了随时排序算法 SAR。该算法能够在处理完指定数量的倒排项后或给定时间内提前终止查询过程，大大减少查询评估延迟时间。

在 SAR 算法实现的基于影响力排序索引中，文

档标识符按照每个词对于文档的实际贡献得分分段，每段以文档标识符升序排列，而段按照影响力分数降序进行排列，最终将影响力分数的 top-k 结果返回。

3.1　影响力分数与词项权重量化

定义 8　影响力分数（impact score，简称 IS）。给定查询 q，文档 d，那么查询 q 中文档 d 的影响力分数可表示为

$$IS_{(d,q)}=\sum_{t\in d\cap q}\omega_{(d,t)}\cdot\omega_{(q,t)},\qquad(2)$$

其中：$\omega_{(d,t)}$ 为词项 t 对于文档 d 的权重，在索引建立过程中被量化到 b 字节中，在 SAR 算法中设置为 8；$\omega_{(q,t)}$ 为词项 t 对于查询词 q 的权重。

对于词项的量化标准，SAR 算法采用了由 Anh 等[37]提出的量化方法：

$$I_{(d,t)}=\left\lfloor\frac{\omega_{(d,t)}-M_{\min}\omega_{(d,t)}}{M_{\max}\omega_{(d,t)}-M_{\min}\omega_{(d,t)}}\times 2^b\right\rfloor。\qquad(3)$$

3.2　索引的组织方式

索引的组织方式如下，单词字典中的每个查询词项指向倒排列表，倒排列表中的倒排项由类似{score，start，end，num}的四元组组成，称之为段（segment）。其中段的第一项 score 代表影响力分数，第二项 start 代表指向段数据首部的指针，第三项 end 代表指向段数据尾部的指针，包含在段数据中的文档数量则由变量 num 存储。每个词项的段都按照以段中存储的 score 值降序、文档标识符升序排列。

3.3　SAAT 评估策略

基于以上影响力分数计算以及索引组织方式，应用查询评估策略 SAAT。在 SAAT 查询处理机制的剪枝方法中，定义了 4 种查询处理模式：

定义 9　OR 模式。在该模式下，所有文档都将分配分数累加器，且都会进行得分统计。

定义 10　AND 模式。若转换为该模式，则出现的新文档不再被分配分数累加器，只针对已被分配累加器的文档进行分数累计操作。

定义 11　REFINE 模式。该模式应用的前提是，top-k 的文档已经确定，但最终顺序还未确定。此时，得分累加只针对 top-k 的文档。

定义 12　IGNORE 模式。在该模式下，停止对所有文档的得分进行递加，查询处理过程终止。

首先获取与查询词项相关的倒排列表段，然后根据段中存储的 score 值进行降序排列，并按照此顺序对段进行处理。对于段中的每个文档标识符，其影响力分数值由文档对应的累加器存储，而在处理过程中累加器中的值通过维护一个堆来实时获取 top-k 的结果。每当将当前影响力分数值添加到累加器时，通过与堆顶值进行判断可决定是否将指向累加器的指针添加到堆中。

由于实时地维护了影响力值最大的 top-k 个文档结果，因此能够在任意给定时间或给定处理倒排列表项的数量终止算法，返回给用户检索结果。另外，段会按照优先度依次递减的顺序处理，优先度由词项的分数贡献值决定，因此排名情况会随着查询进展逐步细化。若查询时间预算增加，则输出结果的质量也成正比提升。

3.4　提前终止阈值参数 η

在处理段的过程中，SAR 算法维护已处理文档影响力得分的累加值。在下一个段处理之前，首先与 η 进行比较，若大于 η 值，则跳出循环，然后从堆中获取 top-k 的结果；若小于 η 值，则流程继续。

基于以上原理介绍，SAR 算法的核心代码如算法 1 所示。

算法 1　SAR 算法

输入：根据影响力得分值进行排序的段遍历器 $L=\{L_1,L_2,\cdots,L_n\}$，返回的文档结果数 k。

输出：影响力得分值 top-k 的文档号及其分数。

1. $AC\leftarrow\{\ \}$；$T\leftarrow\{\ \}$；$RF\leftarrow\{\ \}$；mod←OR；
2. 　计算每个词项 t 的 npb_t；
3. 　for $i\leftarrow k^2$；$i>0$；$i\leftarrow i-1$ do
4. 　for each $\langle t,d,\omega_{(d,t)}\rangle$ in block $i=\omega_{(d,t)}\cdot\omega_{q,t}$ do
5. 　　　if mod=OR or $AC_d>0$ then
6. 　　　$AC_d\leftarrow AC_d+i$；$T_d\leftarrow T_d\cup\{t\}$；
7. 　更新 npb_t；
8. 　　　if Score$\geqslant\sum\{npb_t\mid t\in q\}$ then
9. 　　　　　mod←AND；
10. 　　if Score$\geqslant$max$\{MAX_d\mid d\in AC,d\notin R\}$ then
11. 　　　mod←REFINE；
12. 　　　for each $d\in AC$ and $d\notin R$ do
13. 　　　$AC\leftarrow AC-AC_d$；
14. 　　if $AC_j\geqslant MAX_j$，$(j\in R)$ then
15. 　　　　break；
16. 　return 累加器中得分最高的 k 个文档的文档号。

SAR 算法核心代码如算法 1 所示。步骤 1 使用 OR 模式对各个查询词项对应倒排表中分数高的段进行处理；步骤 2～11，计算每个词项 t 对应倒排表中未处理块的最大分数，即 npb_t。当文档得分大于 npb_t 时，将 OR 模式改用 AND 模式；步骤 12～13，若文档得分大于所有文档的最大得分，即满足条件

Score≥max{MAX_d | $d \in AC, D \notin R$}时，将模式改用 REFINE 模型进行处理。其中，AC 为现有累加器集合，保存文档号及文档的部分得分，M_d 为文档 d 的最大得分，由 AC 保存的分数累加得到，即 $MAX_d = AC_d + \sum\{npb_d | t \in q, t \notin T_d\}$；步骤 14～15，若满足现有累加器集合中的累加分数大于文档 d 的最大分数，则此时查询可以提前终止，采用 IGNORE 模式。最终得到累加器中得分最高的 top-k 个文档。

4 实验评估

对 APS 算法和 SAR 算法分别进行评估。

针对 APS 算法，使用误差计数法对 APS 算法以及优化前的波特词干算法进行评估，利用该方法通过计算词干提取不足指数(understemming index，简称 UI)、词干提取过度指数(overstemming index，简称 OI)以及相对截断错误率(error rate relative to truncation，简称 ERRT)3 个指标对 APS 算法的词干提取准确率进行评价，最后在 2 个数据样本上进行实验验证，并与现有词干算法进行对比。

针对 SAR 算法，在 2 个真实的大型 TREC 标准数据集上进行实验验证，通过检索质量评价指标 nDCG@10 对 SAR 算法进行评估，并说明了在给定时间预算下的查询延迟、减少的倒排段处理数量等。

4.1 实验环境及数据集

实验的硬件环境为 Intel® Xeon® CPU E3-1226 v33.30 GHz 和 256 GiB 内存；软件环境为 Red Hat Enterprise Linux 6。

针对 APS 算法的评估，实验在 2 个真实数据集上开展，数据集基本信息如下：

1)Word List A：来自于 Paice 官方网站，最初用于 Paice 评估，包含约 10 000 个词。词汇样本取自于图书情报学相关的 CISI 测试集。

2)Word List B：由 Scrabble 单词检查器中使用的单词列表编译而成，该样本包含约 20 000 个单词。

针对 SAR 算法的评估，实验在 2 个标准 TREC 测试集 ClueWeb09、数据集 ClueWeb12-B13 进行。通过检索质量评价指标 nDCG@10 对 SAR 算法进行评估。数据集的文档数量和实验所用到的 TREC 主题如表 7 所示。

表 7 TREC 数据集及主题

数据集	文档数量	TREC 主题
ClueWeb09b	50 220 189	51～200
ClueWeb12-B13	52 343 021	201～300

另外，本实验对数据集中的每个文档进行了如下处理：将所有无效 UTF-8 字符转换成了空格，同时对字母字符与数字字符进行分离，并剔除了标记标签。

4.2 APS 算法评价

在 2 个数据集样本上对 APS 算法进行实验。首先，为了形成对照，将改进后的 APS 算法与改进前的 Porter Stemmer 算法进行评估对比；之后，在数据集上对现有的词干分析算法 Paice/Husk 及 Lovins 也进行了对比测试，作为数据参考。通过实验验证得知，与现有词干分析算法相比，APS 算法提高了对查询词词干提取的准确率，实验结果如图 2 所示。

以 Word List A 数据样本为观察对象，图 2(b)、(c)中，APS 算法与改进前的波特词干算法相比，词干不足指数 UI 降低了约 48.4%，相对截断错误率 ERRT 降低了约 28%。UI 值的改善说明 APS 算法能对更多相关词合并成同一词干，例如对于单词“ability”和“able”的处理，改进前的波特词干算法并不会将其归为同一词干群。图 2(a)中 OI 值之所以相对改进前有所提升，是因为 APS 算法调整规则函数后删除了许多重要词缀，这对 OI 值造成了影响。实际上 UI 值的改善会在一定程度上影响 OI 值，导致词干提取过度，但影响的单词数较少。因此，根据 ERRT 值对总体相对准确性的评估来看，APS 算法对于词干提取的效果要优于波特词干算法。

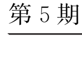
以 Word List B 数据样本作为观察对象。由图 2(e)、(f)可知，APS 算法较改进前，词干不足指数 UI 降低了约 54.6%，相对阶段错误率 ERRT 降低了约 30.2%。可以发现，在 Word List B 数据样本中，APS 算法对于词干提取的准确率具有较大的提升，能够将更多的相关词统一成同一词干。

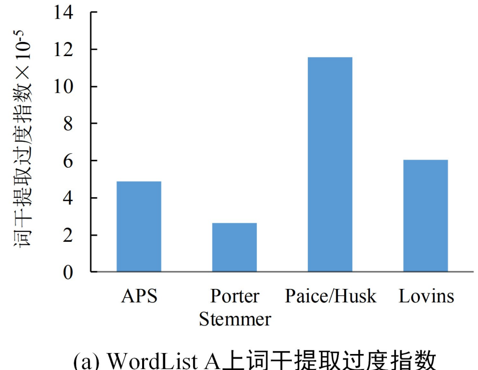
除此之外，通过和 Lovins、Paice/Husk 算法对比可知，APS 算法表现更佳，其中相对截断错误率的数据表明，APS 算法相对于其他的词干提取算法，有效提升了词干提取准确率。

4.3 SAR 算法评价

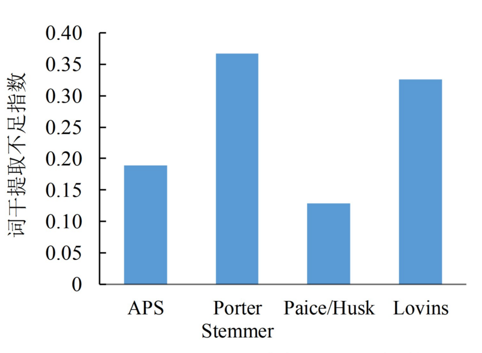
对于 2 个评价数据集，将前十个主题用于训练线性模型，其余主题用于测试。评价效率的指标只包括引擎框架生成 top-k 结果花费的时间，即查询延迟时间，不包括将单词字典、倒排列表加载到主存储器的启动成本以及写入输出文件的时间。查询延迟时间通过 chrono 库进行测量，检索质量选用 nDCG@10 作为度量指标。

通过将倒排项数量 η 分别设置为 10^4、10^5、10^6、

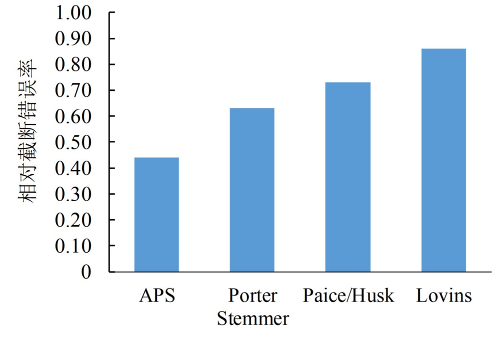




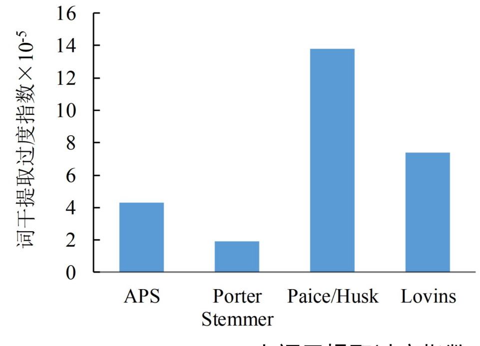
(a) WordList A上词干提取过度指数



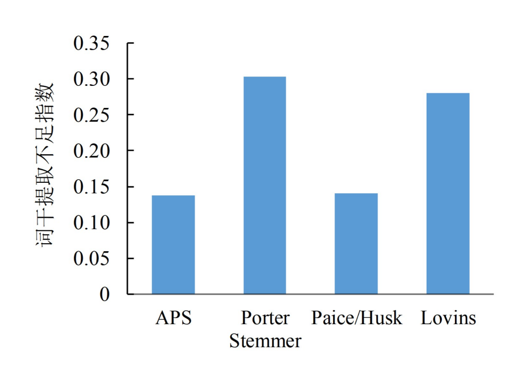
(b) WordList A上词干不足指数



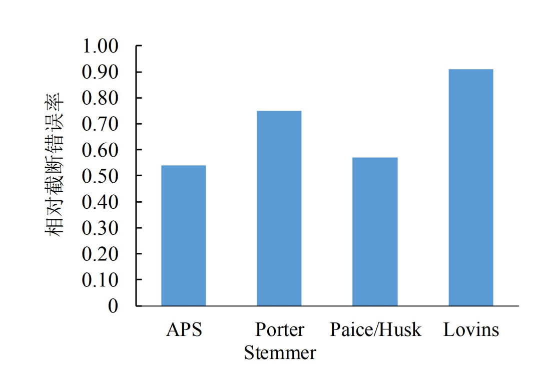
(c) WordList A上相对截断错误率



(d) WordList B上词干提取过度指数



(e) WordList B上词干提取不足指数



(f) WordList B上相对截断错误率

图 2　APS 算法词干提取准确率评价

10^7 以及 10^8 观察 nDCG@10 的变化,从而确定倒排项数量 η 的最佳取值。图 3 为在给定处理倒排项数量 η 变化时,nDCG@10 指标的变化情况。由图 3 可知,在不显著影响检索质量的情况下,SAR 算法有效减少了需要处理的倒排段数量。通过分析折线趋势可以发现,将 η 设置为数据集大小的 10%最为合理,因为在 $\eta=10^7$ 与 $\eta=10^8$ 时,指标 nDCG@10 数据表现效果不相上下。由上一步分析得到 η 最佳取值范围后,在此基础上用 2 个测试集合 ClueWeb09b 和 ClueWeb12-B13 的前 10 个主题训练模型,记录在给定时间预算的情况下,查询的延迟时间和处理的倒排段数量。由此模型来预测在给定时间预算下 η 的最佳取值。数据集 ClueWeb09b 和数据集 ClueWeb12-B13 符合线性回归的特点,其线性模型包括恒定的开销和每个倒排段的处理成本。通过最终的线性模型,确定 η 适当的取值后,将时间预算分别设置为 25、50、100、150、200 ms。在此条件下进行 3 次测试取平均值,最终 SAR 算法在 2 个数据集上的检索质量如图 4 和图 5 所示。

图 4 和图 5 中 max 取值由双侧配对随机化测验得到,并作为标准值来体现相对有效性差异。由图 4 和图 5 可看出,在给定时间预算下,SAR 算法检索质量有一定程度的下降,但在可接受范围内;由图中折线的总体趋势可以发现,随着给定预算时间的延迟,

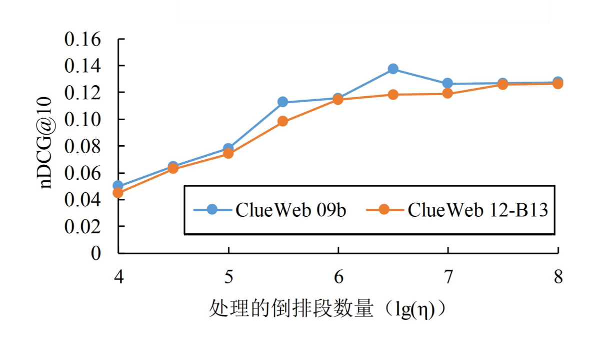


图 3　给定倒排项数量 η 时的 nDCG@10 指数

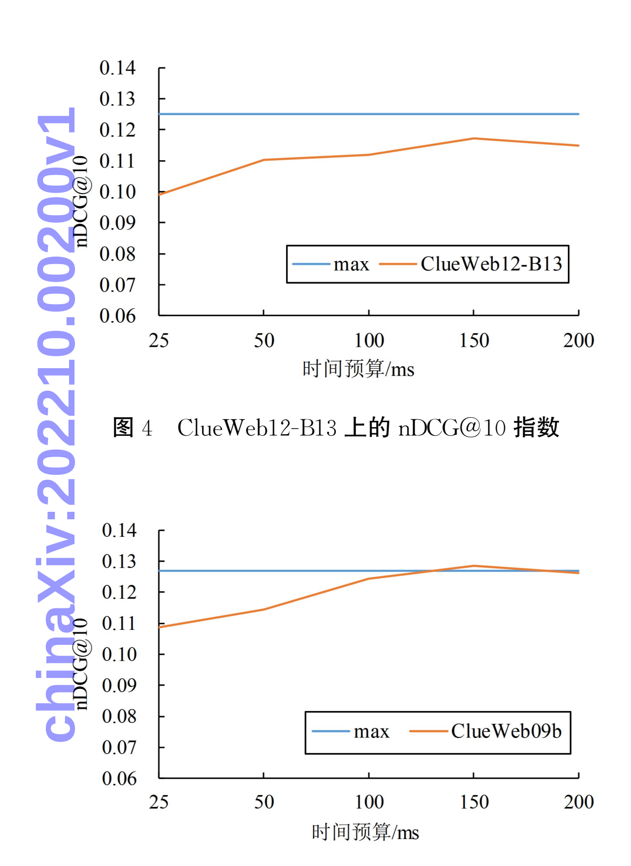


图 4　ClueWeb12-B13 上的 nDCG@10 指数

图 5　ClueWeb1209b 上的 nDCG@10 指数

检索质量也相应提升。另外，由 2 个图的数据对比可知，在数据集 ClueWeb12-B13 上处理所有倒排项所花费的时间要比数据集 ClueWeb09b 要长，这说明在相同的时间预算下，数据集越大，有效性折损也越大，因此，ClueWeb12-B13 的 nDCG@10 指标折损更多。

图 6 和图 7 为在 2 个数据集上的平均延迟时间，图 8 和图 9 为在 2 个数据集上的提前终止倒排段的数量与倒排段总数量。由图 6～9 可知，SAR 算法通过在给定查询时间内提前终止查询过程，大大减少了倒排项的处理数量，从而有效减少了查询延迟时间。

表 8 和表 9 为 2 个数据集上未处理的主题数与给定查询时间下的超时时间。

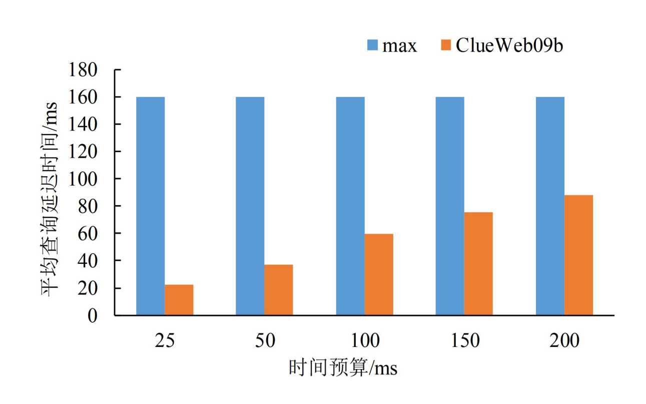


图 6　ClueWeb12-B13 上的平均查询延迟时间

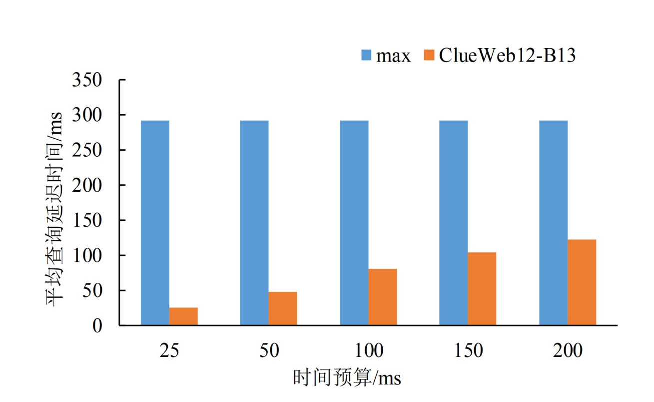


图 7　ClueWeb09b 上的平均查询延迟时间

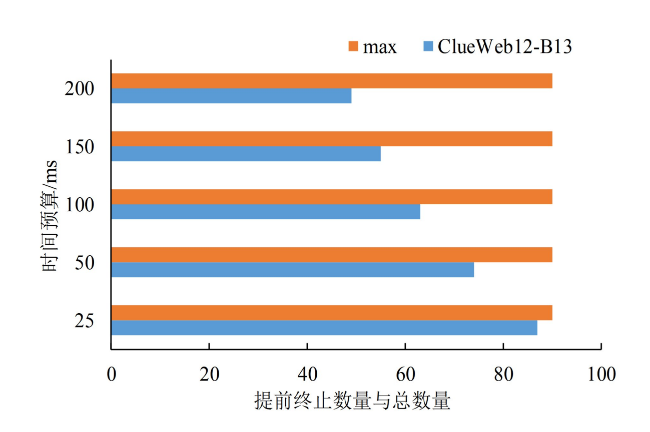


图 8　ClueWeb12-B13 上的提前终止数量与总数量

由上述实验结果分析可知，SAR 算法在特殊情况下存在略微的延迟，总体来看影响并不大，但在控制查询延迟时间方面效果显著。另外，随着预算时间的增加，检索质量也相应成正比提升，虽然存在一定程度的检索质量下降，但在可接受的范围内。实验结果也验证了 SAR 算法对控制尾部延迟的有效性，能够减少计算资源的消耗，且对于用户体验的提升也有一定帮助。

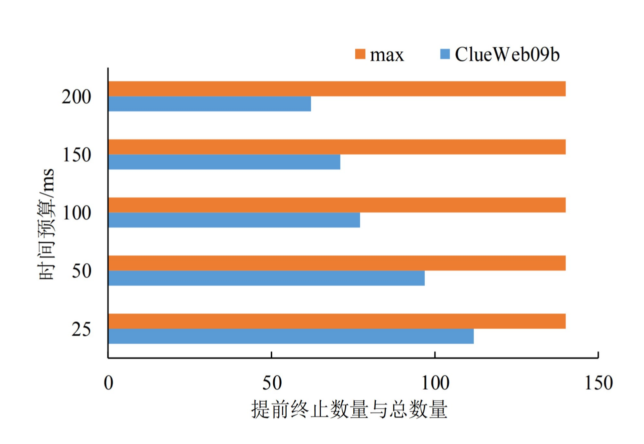


图 9 ClueWeb09b 上的提前终止数量与总数量

表 8 ClueWeb09b 数据集上未处理主题数与超时时间

预算时间/ms	未处理主题数	平均超时时间/ms	最大超时时间/ms
25	47	0.78	2.3
50	3	0.75	1.5
100	0	0	0
150	0	0	0
200	0	0	0

表 9 ClueWeb12-B13 数据集上未处理主题数与超时时间

预算时间/ms	未处理主题数	平均超时时间/ms	最大超时时间/ms
25	74	0.63	1.2
50	46	1.2	3.4
100	0	1.8	3.7
150	0	0	0
200	0	0	0

5 未来展望

基于 APS 算法对文本预处理进行了优化,并基于 SAAT 策略设计了随时排序算法 SAR,在数据集上的实验结果达到了预期的效果,但考虑到时代环境的需求变化以及对各种场景的适用情况,该检索系统的扩展未来还有一定的优化空间,需要相关的研究和工作支持。为此,从几个方面提出了需要进一步研究与探讨的工作点:

首先,针对倒排索引,可以考虑利用数据压缩算法对其进行压缩,以减少索引占用的磁盘空间,进而降低磁盘读写数据的时间开销。在之后的工作中可以在该检索系统中添加一个简单有效的解编码器,例如基于单指令多数据流(single instruction multiple data,简称 SIMD)的解编码器[38-39],将压缩和解压的过程并行化,以实现存储空间的减少和访问速度的提升。

其次,由于文档长度存在不确定性,词频存在随机性,为提高对文档中稀有词项的建模能力,实现带有 Dirichlet 平滑(dirichlet smoothing,简称 DiS)方法或 JM 平滑方法(jelinek-mercer smoothing,简称 JMS)的语言模型[40]也是可行的优化点之一。对文档和查询项进行语言建模后,不仅能够提高估计文档语言模型的准确性,而且也能适应查询中非常用词的生成。

最后,可以针对用户接口设计更利于用户体验的界面。目前本文检索系统的接口尚且基于文本,后期可以通过 HTML 界面来实现用户交互接口。用户在界面展示的文本框中输入查询词后,搜索的结果能够通过该界面进行展示以供阅读、分析和判断。对交互接口进行优化能够丰富表现信息的形式,便于用户多方式高效接收信息,从而进一步提升用户体验。

6 结束语

针对文本预处理阶段,设计了优化的词干分析算法 APS,基于派生形态学调整了规则函数的定义,改善了波特词干算法存在的词干提取不足以及准确率不理想的问题,并通过实验验证了 APS 算法在提升词干提取准确率的有效性。另外,针对相关排序阶段,基于 SAAT 查询策略设计了随时排序算法 SAR,能够在给定时间预算或给定处理的倒排段数量的情况下,提前终止检索过程,减少不必要的时间消耗,有效控制查询延迟,返回较为准确的检索结果。在 2 个大规模 TREC 数据集上的实验结果验证了 SAR 算法对于控制尾部延迟时间的有效性。最后,本文提出了若干可行的研究点,为未来的工作指明了方向。

参考文献:

[1] ADADI A. A survey on data-efficient algorithms in big data era[J]. Journal of Big Data, 2021, 8(1): 1-54.

[2] BRUTLAG J. Speed matters for Google web search [EB/OL]. (2009-06-23)[2012-02-15]. https://venturebeat.com/wp-content/uploads/2009/11/delayexp.pdf.

[3] FRIED J, RUAN Z, OUSTERHOUT A, et al. Caladan: mitigating interference at microsecond timescales [C]//Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). Berkeley: CA Press, 2020: 281-297.

[4] ARAPAKIS I, BAI X, CAMBAZOGLU B B. Impact of response latency on user behavior in web search[C]// Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2014: 103-112.

[5] KOHAVI R, DENG A, FRASCA B, et al. Online controlled experiments at large scale[C]//Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY: ACM Press, 2013: 1168-1176.

[6] HWANG S W, KIM S, HE Y, et al. Prediction and predictability for search query acceleration[J]. ACM Transactions on the Web, 2016, 10(3): 1-28.

[7] JEON M, KIM S, HWANG S W, et al. Predictive parallelization: taming tail latencies in web search[C]//Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2014: 253-262.

[8] MACKENZIE J, SCHOLER F, CULPEPPER J S. Early termination heuristics for score-at-a-time index traversal[C]//Proceedings of the 22nd Australasian Document Computing Symposium. New York, NY: ACM Press, 2017: 1-8.

[9] YUN J M, HE Y, ELNIKETY S, et al. Optimal aggregation policy for reducing tail latency of web search[C]//Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2015: 63-72.

[10] 任睿，马久跃，隋秀峰，等. 一种减少长尾延迟的分布式实时约束传播方法[J]. 计算机研究与发展，2017，54(7)：1617-1628.

[11] KIM S, HE Y, HWANG S W, et al. Delayed-Dynamic-Selective(DDS) prediction for reducing extreme tail latency in web search[C]//Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. New York, NY: ACM Press, 2015: 7-16.

[12] TROTMAN A, CRANE M. Micro-and macro-optimizations of SaaT search[J]. Software: Practice and Experience, 2019, 49(5): 942-950.

[13] LOVINS J B. Development of a stemming algorithm[J]. Mechanical Translation and Computational Linguistics, 1968, 11(1/2): 22-31.

[14] DAWSON J L. Suffix removal and word conflation[J]. ALLC Bulletin, 1974, 2(3): 33-46.

[15] PAICE C D. Another stemmer[C]//Proceedings of the ACM SIGIR Forum. New York, NY: ACM Press, 1990, 24(3): 56-61.

[16] PORTER M F. Snowball: a language for stemming algorithms[EB/OL]. (2001-07-16)[2022-02-15]. http://snowball.tartarus.org/texts/introduction.html.

[17] PORTER M F. An algorithm for suffix stripping[J]. Program, 1980, 14(3): 130-137.

[18] KROVETZ R. Viewing morphology as an inference process[J]. Artificial Intelligence, 2000, 118(1-2): 277-294.

[19] MAYFIELD J, MCNAMEE P. Single n-gram stemming[C]//Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. New York, NY: ACM Press, 2003: 415-416.

[20] MELUCCI M, ORIO N. A novel method for stemmer generation based on hidden Markov models[C]//Proceedings of the Twelfth International Conference on Information and Knowledge Management. New York, NY: ACM Press, 2003: 131-138.

[21] MAJUMDER P, MITRA M, PARUI S K, et al. YASS: Yet another suffix stripper[J]. ACM Transactions on Information Systems, 2007, 25(4): 18-24.

[22] BONUS D E. The tagalog stemming algorithms (TagSA)[C]//Proceedings of the Natural Language Processing Research Symposium. Manila: DLSU Press, 2003: 272-274.

[23] SILVA G, OLIVEIRA C. A lexicon-based stemming procedure[C]//Proceedings of the International Workshop on Computational Processing of the Portuguese Language. Berlin: Springer Press, 2003: 159-166.

[24] AL-SHAMMARI E T, LIN J. Towards an error-free Arabic stemming[C]//Proceedings of the 2nd ACM Workshop on Improving Non English Web Searching. New York, NY: ACM Press, 2008: 9-16.

[25] STROHMAN T, TURTLE H, CROFT W B. Optimization strategies for complex queries[C]//Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2005: 219-225.

[26] BRODER A Z, CARMEL D, HERSCOVICI M, et al. Efficient query evaluation using a two-level retrieval process[C]//Proceedings of the Twelfth International conference on Information and Knowledge Management. New York, NY: ACM Press, 2003: 426-434.

[27] DING S, SUEL T. Faster top-k document retrieval using block-max indexes[C]//Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. New York, NY: ACM Press, 2011: 993-1002.

[28] DIMOPOULOS C, NEPOMNYACHIY S, SUEL T. Optimizing top-k document retrieval strategies for block-max indexes[C]//Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. New York, NY: ACM Press, 2013: 113-122.

[29] MALLIA A, OTTAVIANO G, PORCIANI E, et al. Faster BlockMax WAND with variable-sized blocks

[C]//Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2017: 625-634.

[30] CRANE M,CULPEPPER J S,Lin J,et al. A comparison of document-at-a-time and score-at-a-time query evaluation[C]//Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. New York,NY:ACM Press,2017:201-210.

[31] KIM Y,CALLAN J,CULPEPPER J S,et al. Efficient distributed selective search[J]. Information Retrieval Journal,2017,20(3):221-252.

[32] KULKARNI A,CALLAN J. Selective search: efficient and effective search of large textual collections[J]. ACM Transactions on Information Systems, 2015, 33(4):1-33.

[33] LIN J,TROTMAN A. Anytime ranking for impact-ordered indexes[C]//Proceedings of the 2015 International Conference on the Theory of Information Retrieval. New York,NY:ACM Press,2015:301-304.

[34] GOODWIN B,HOPCROFT M,LUU D,et al. BitFunnel: revisiting signatures for search[C]//Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York,NY:ACM Press,2017:605-614.

[35] PEDERSEN J. Query understanding at Bing[EB/OL]. (2012-03-15)[2022-02-15]. https://dl. acm. org/doi/pdf/10. 1145/2009916. 2010190.

[36] TROTMAN A, JIA X, CRANE M. Towards an efficient and effective search engine[C]//Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval. Dunedin:University of Otago,2012:40-47.

[37] ANH V N, DE KRETSER O, MOFFAT A. Vector-space ranking with effective early termination[C]//Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM Press, 2001: 35-42.

[38] PIBIRI G E, VENTURINI R. Techniques for inverted index compression [J]. ACM Computing Surveys, 2020,53(6):1-36.

[39] TROTMAN A. Compression,SIMD,and postings lists [C]//Proceedings of the 2014 Australasian Document Computing Symposium. New York, NY: ACM Press, 2014:50-57.

[40] ZHAI C X, LAFFERTY J. A study of smoothing methods for language models applied to information retrieval[J]. ACM Transactions on Information Systems,2004,22(2):179-214.

编辑:梁王欢